Intrusion Detection in Industrial Networks via Data Streaming

Downloaded from: https://research.chalmers.se, 2023-05-05 17:29 UTC

Citation for the original published paper (version of record):

Butun, I., Almgren, M., Gulisano, V. et al (2020). Intrusion Detection in Industrial Networks via Data Streaming. Industrial IoT: Challenges, Design Principles, Applications, and Security: 213-238. http://dx.doi.org/10.1007/978-3-030-42500-5_6

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004. research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

Intrusion Detection in Industrial Networks via Data Streaming

Ismail Butun (Ph.D.), Magnus Almgren (Ph.D.), Vincenzo Gulisano (Ph.D.) and Marina Papatriantafilou (Ph.D.)

Abstract Given the increasing threat surface of industrial networks due to distributed, Internet-of-Things (IoT) based system architectures, detecting intrusions in Industrial IoT (IIoT) systems is all the more important, due to the safety implications of potential threats. The continuously generated data in such systems form both a challenge but also a possibility: data volumes/rates are high and require processing and communication capacity but they contain information useful for system operation and for detection of unwanted situations.

In this chapter we explain that stream processing (a.k.a. data streaming) is an emerging useful approach both for general applications and for intrusion detection in particular, especially since it can enable data analysis to be carried out in the continuum of edge-fog-cloud distributed architectures of industrial networks, thus reducing communication latency and gradually filtering and aggregating data volumes. We argue that usefulness stems also due to facilitating provisioning of agile responses, i.e. due to potentially smaller latency for intrusion detection and hence also improved possibilities for intrusion mitigation. In the chapter we outline architectural features of IIoT networks, potential threats and examples of state-of-the art intrusion detection methodologies. Moreover, we give an overview of how leveraging distributed and parallel execution of streaming applications in industrial setups can influence the possibilities of protecting these systems. In these contexts, we give examples using electricity networks (a.k.a. Smart Grid systems).

We conclude that future industrial networks, especially their Intrusion Detection Systems (IDSs), should take advantage of data streaming concept by decoupling semantics from the deployment.

Pre-print

Ismail Butun, Magnus Almgren, Vincenzo Gulisano, and Marina Papatriantafilou
Chalmers University of Technology, Department of Computer Science and Engineering (CSE), Network and Systems Division, 412 96 Gothenburg, Sweden,
emails: {ismail.butun, magnus.almgren, vincenzo.gulisano, ptrianta}@chalmers.se
Acknowledgement: This research has been partially supported by the Swedish Civil Contingencies Agency (MSB) through the projects RICS, by the EU Horizon 2020 Framework Programme under grant agreement 773717, and by the Swedish Foundation for International Cooperation in Research and Higher Education (STINT) Initiation Grants program under grant agreement IB2019-8185.

1 Introduction

The digitalization of society is on-going and affects everything from consumers and their home devices/networks to complex industrial systems such as the *smart grid*. The digitalization brings many advantages, but also comes with risks: due to distributed Internet-of-Things (IoT) architectures, the threat surface is rapidly increasing and many societal critical systems have become susceptible to cyber attacks and even attacked. Anecdotally, whereas cyber attacks and defenses of the 20th century focused on data and information, attacks of the 21st century also focus on physical manipulation of systems with devastating effects [15].

Even though cyber security for traditional IT systems is quite mature, there is a gap when it comes to deploying security mechanisms for industrial networks. The architecture is highly distributed, data is generated at many points in the system and the aggregated volume at central nodes is very big even if data is highly filtered and aggregated beforehand (which may in turn remove important indications of attacks). Orthogonal, there are also aspects of confidentiality and privacy of data which argue for some analysis to be local to the node where data is generated while a full system attack analysis needs to be done centrally.

The explosion of sensor data in the last twenty years has also shown the limitations of databases, where information first needs to be stored and then queried. Driven by general need for analysis of very large data, the *streaming paradigm* was developed emphasizing the *query/analysis* of the data over data persistence in the form of databases. Considering data as flows between nodes makes the data streaming paradigm well suited to the analysis of data in highly distributed industrial networks, such as the smart grid. Data is analyzed on the node and then streamed throughout the system. However, even with the advantages of data streaming for general analysis of data, this same paradigm has not been widely adopted by the security community to enhance security mechanisms tailored to industrial networks.

Following the earlier chapters' presentations of wireless communication needs and challenges, automation trends and applications of Industrial IoT (IIoT) and industrial networks, this chapter is dedicated to present *intrusion detection* and how it can be combined with the principles from *data streaming* to build more efficient attack detection systems for industrial networks.[1]

As data is generated at high rates and volumes in the industrial networks, bottlenecks occur from data/command source to destination, which eventually increases the processing and response times of the queries/commands. In this aspect, we project that the *data streaming paradigm would be a remedy* in helping the efforts of migration of cloud to the edge, meaning that the data are first analyzed as close to the source as possible, before a subset / refinement is sent elsewhere for further analysis. Apart from reducing the volume of data to be analyzed centrally, the local analysis is also faster, meaning that if early signs of attacks are detected the node can quickly reconfigure itself to mitigate the attack.

[1] Industrial networks and IIoT are used interchangeably, with a preference for the former term.

Pre-Print

The rest of the chapter is structured as follows: Section 2 presents the preliminaries by introducing the distributed network architecture of evolving industrial networks along with our use-case scenario, the smart grid. We look at requirements suggested in literature for such networks, as well as give a summary of actual attacks to learn from when designing new detection algorithms. We then introduce *data streaming* in Section 3 with a short motivating attack detection example. Section 4 introduces IDSs, with their underlying detection algorithms with some concrete examples. In Section 5, we discuss how streaming-based applications can be leveraged to improve intrusion detection systems. Finally, Section 6 concludes the chapter.

2 Preliminaries

This section gives an introduction to the distributed network architecture of evolving industrial networks, followed by an example of such infrastructures, namely a generic smart grid system. The security requirements in such networks are discussed as well as important historical attacks found in the literature.

2.1 Distributed edge-fog-cloud architectures

Nowadays, with the adoption of edge-computing, the computing resources are approaching to the end-devices from the cloud, hence the term "distributed network" is widely used. We foresee that future industrial network deployments will benefit from this distributed networks wisely. An example to evolving multi-tier architecture of IIoT, Fig. 1 [42] presents an abstraction, consisting of:

- Cloud / high-end layer, where processing devices are commonly high-end servers, multi/many -core systems, or supercomputers, such as a *data center* of a business network, in which final data allocations, processing, storage, etc. are handled. Location wise, they are at the farthest point of the network from the low-end layer, yet reachable to/from each end-device.
- Fog / intermediate layer, where processing devices provide moderate computing power, such as edge optimized network servers, located in the vicinity of the low-end layer, in order to provide real-time experiences to the low-end devices.
- Edge / IoT / low-end layer, where resource-constrained on-premise devices are representative examples, such as machines, robot arms, door locks, sensors, etc.

The fog layer constitutes an intermediate layer in between IoT and cloud layers, to provide ways of leveraging cloud layer resources by the resource-constrained IoT end-devices.

Pre-print

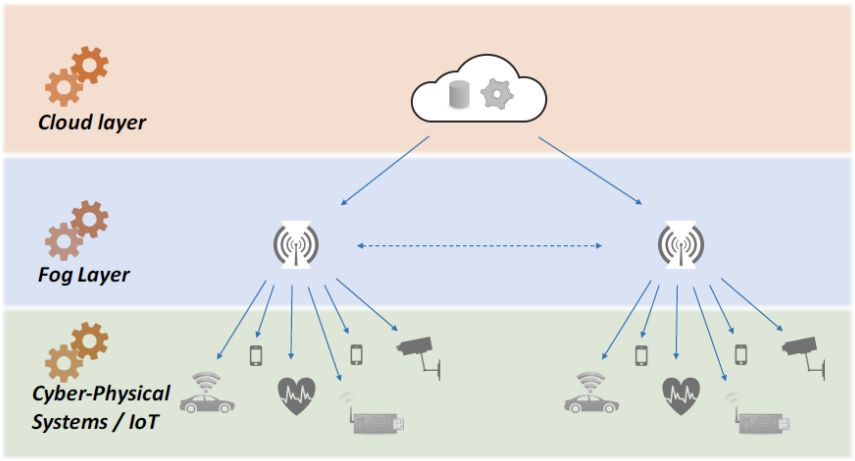

Fig. 1: Fog computing and cloud computing layers in IoT networks [42].

2.2 A use-case scenario of an industrial network: A smart grid system

As a use-case scenario of an industrial network, we present a smart grid system. A full grid contains the generation, transmission and the distribution, where we emphasize the electricity distribution network with the coupled information network on top. This part of the grid contains the infrastructure for distributing electricity to customers, from the transmission grid out to the substations and finally the customers with smart meters to measure quality of the delivered electricity as well as to handle billing. The smart meters can be seen as a type of IoT devices, which communicate using ZigBee, BLE, LoRa, NB-IoT, etc. All together, they build up the so called Advanced Metering Infrastructure (AMI), which is the heart of a smart grid system. Typically, we can distinguish the following layers of a smart grid system:

1. **Smart meters:** Instant usage data from the smart meters are collected by the sensors and then sent to the utility head-ends via data concentrators, for further analysis and/or storage to be used at billing, statistics, etc. Think of *IoT layer* of the Fig. 1.
2. **Data concentrator:** Data from many (hundreds-thousands) smart meters are collected at the data concentrators and relayed to the AMI head-end. Think of *Fog layer* of the Fig. 1.
3. **AMI head-end:** A broad spectrum of the data is stored and processed at the AMI head-end (the utility side), such as the consumer short/long term electricity usage data. Think of *Cloud layer* of the Fig. 1.

As a specific example; electricity usage data of the consumers follow the path of 1-2-3 (upstream); whereas commands (such as pushing software patches) from the control center follow the reverse path of 3-2-1 (downstream).

Pre-print

Some data generated at the smart meter is used for billing. As such, it needs to be correct, can be aggregated and only available to the AMI head-end at regular intervals and not in real time. Other data generated at the smart meter offers the possibility to improve the overall electricity quality if it can be shared instantaneous with at least the data concentrator (and the corresponding sub station for the electricity flow). For the best effects, this data should be sampled often (implying customer privacy concerns) but it would not need to be stored.

Considering the attack surfaces of the system, attacks can target any level of the system and the communication in-between. As such, analysis should include data from the individual smart meters (analyze number of local failed logins), some aggregate at intermediate nodes from smart meters to understand if some behavior is wide-spread (a wide-spread attack to many nodes to probe for weak passwords), with the final aggregate analysis in the head-end.

2.3 Requirements for protecting industrial networks from cyber-attacks

Kumar et al. list the following basic requirements for protecting the users of industrial networks and also the network itself from cyber-attacks [28].

For the network:

- **Integrity of data and commands:** The integrity of the data is critical, as it might affect the operation functions in the factories, the AMI meter readings, the sent commands to the actuators, etc.
- **Availability against DoS/DDoS attacks:** Denial-Of-Service (DoS) attacks cause depletion of the network resources (e.g. throughput, bandwidth, etc.) by sending fake requests either to the server or to the whole network. On the other hand, Distributed DoS (DDoS) attacks are executed by leveraging distributed captured components, such as compromised smart meters of the AMIs; captured firewalls, routers and switches of regular networks; and/ or consumer appliances and using them against a single target. DDoS is one of the most dreadful attacks against industrial networks and it is hard to circumvent [47]. For instance, availability of the pricing information and also the power are the key aspects of smart grid networks, therefore they need to be somehow guaranteed (pricing for the provider, and power for the consumer).

For the users:

- **Confidentiality of the usage:** The data including the usage of the services offered by the industrial network provider should be kept confidential. For instance, in a smart grid network, short/long -term electricity usage information of a commercial customer, i.e. an industrial company, should be kept confidential, in order to protect company's production secrets from industrial espionage.

Pre-print

- **Data privacy of the users**: Consumers' private information, such as Personal Identification Information (PII) should not be revealed to the outsiders; an adversary shall not gain any knowledge about individual users of the industrial network without the will of the consumers. Among PII, most sensitive ones are citizen ID number, passport number, passwords of online banking accounts, credit card information, or other financial data, etc. [12]. For the case of smart grid networks, the information pertaining the instant electricity usage patterns might be sensitive as they may reveal personal activities such as presence at the home, being awake or not, etc. All these kinds of sensitive user information should be protected by the network operator (for instance, electricity utility provider in smart grid systems) from unauthorized viewers. More importantly, future implementations of the smart grid systems should comply with the newly released General Data Protection Regulation (GDPR, privacy law of E.U.), by informing the consumers about their data collection processes and obtaining their consent; along with applying transparent data storage/processing policies [37]. Therefore, the utility center should be aware of the total consumption information for billing operations, yet should decide storing the details of the daily consumption pattern of the individuals according to their privacy choice [3].

Privacy and Confidentiality Goals: As mentioned above, confidentiality of the industrial customers, and privacy of the individual consumers are very important for the industrial networks. Especially after the GDPR law, privacy carries prime importance and if violated, it might be costly for the network operators (such as the utility providers in smart grid systems). While building the industrial networks, particularly following enlisted privacy and confidentiality goals should be aimed at by the network operators [27]:

- *Anonymity*: A user should not be identifiable within a set of subjects.
- *Unlinkability*: After the billing service, any consumption data should not be linkable to the related customer.
- *Undetectability*: The consumption data should not be detectable by adversaries.
- *Unobservability*: An outsider should not observe whether the communication takes place or not regarding execution of certain system related messages and/or other actions of interest, such as sending consumption messages, demand-bidding messages, etc.
- *Pseudonymity*: In smart grid communication, many parties may want to have access to the consumption data from the smart meters or AMI, therefore, a smart meter should have a pseudonym identifier. These pseudonym identifiers can only be possessed by the dedicated entities that are communicating or exchanging messages with the smart meter.

Having these requirements in mind, let's now look at actual attacks that are found in the literature.

Pre-print

2.4 Known Cyber-Security Attacks on Industrial Networks

The frequency of cyber-attacks involving industrial networks, especially CPSs, has been expanding. Here, we present only a small sample of well known incidents in chronological order [43]:

Maroochy Shire Sewage Spill (2000:) The city council of Maroochy Shire (Queensland, Australia) has outsourced the water treatment facility automation job to a contractor in 1997 and the contractor installed Supervisory Control and Data Acquisition (SCADA - industrial automation and control standard devised by Siemens Inc.) automation tools to the 142 sewage pumping stations. The number of faults recorded has never exceeded two or three per day till late January 2000. However, the number of faults increased drastically whenever cyber-intrusions happened and continued till the date they were identified on 23 April 2000. The pumping stations were normally controlled by the main SCADA station through the dispersed Remote Terminal Unit (RTU)'s. The sabotage was executed by an old employee who took advantage of the wireless RTU's to execute the manipulating commands, an execution of man-in-the-middle (MITM) attack. This overall incident caused a spill of 264K gallons of raw sewage to the environment, which caused an overall bill of $676,000. This is the first ever reported cyber-security incident in the history of SCADA systems [36].

Slammer (2003): Slammer malware targeted a nuclear power plant located at Ohio in 2003 and shut off its safety monitoring system for five hours. After the incident, it was revealed that the attacker followed a T-1 communication line to connect the corporate network which bypassed the plant's firewall.

Aurora (2007): This vulnerability has shown in 2007 to affect systems that control rotating machinery in the industrial sites such as turbines and diesel generators.

BlackEnergy (2009): This malware targets the Human-Machine Interface (HMI) software of the industrial control systems. It is believed that the first ever hacker-caused power-outage at Ukraine in 2015 was caused by this malware. BlackEnergy was used to steal a legitimate user's Virtual Private Network (VPN) credentials, and by using that attackers gained remote access to the SCADA network of the power distribution and also the HMI. Then they executed further physical attacks such as shutting down the circuits and so on. In a very similar incident at USA, attackers inserted rogue code in software to control electrical turbines of a power plant in 2009.

Stuxnet (2010): This incident is very famous due to political reasons. In 2010, the Stuxnet worm was able to take over many of the Programmable Logic Controllers (PLCs) controlling the centrifuges of the Iranian nuclear facilities, disrupted their centrifuge speed and eventually destroyed them. It has been shown that this worm can be tailored as a platform for attacking smart grid systems that are composed of SCADA systems.

Pre-Print

Vampire attack (2011): Cyber-attacks are sometimes intended to extract secret information, but otherwise, to destroy the communication abilities of the network under attack. For instance, *Vampire Attacks*, in the category of DoS attacks [45], is a very good example of this, in which an attacker intends to drain batteries of the target wireless nodes. In the long run, this type of attack causes a DoS in the overall network; by depleting the batteries of the sensors in the network and causing partition and segregation in the network. These kind of attacks are hard to detect and tough to cope with, and especially dangerous for the IIoT [14].

Havex (2014): It is a malware that uses Remote Access Trojan (RAT) to infiltrate and modify the default software in ICS and SCADA systems. In 2014, it has targeted a number of European companies that develop industrial applications and appliances.

Stealthy attack (2015): Industrial networks are becoming increasingly susceptible to sophisticated and targeted cyber-attacks initiated by attackers with motivation, domain knowledge, and resources. Recently, a specific kind of attack called *Stealthy Attack* has been discovered to be seriously threatening the industrial environments due to the nature of the attack [29]. The adversaries hide their attacks even at the process level, by injecting just enough malicious data that the compromised sensor values still remain approximately within the noise level. Such stealthy integrity attacks are very tough to detect by anomaly detectors that are not sensitive to noise level fluctuations. Solutions to this kind of attack might require specification-agnostic techniques that monitor time series of sensor measurements for structural changes in their behavior [8]. Coping with this kind of attacks is not easy since they require rigorous tests on carefully crafted attacks in a simulation setting by using the prerecorded data-sets from the selected test-beds with a duration of several days.

Mirai (2016) and Torii (2018) botnet attacks: Because of lacking rigid security precautions and bad user habits, IoT devices are leveraged as a workforce of the botnets by ill-mannered hackers. For instance, *Mirai malware* is released against Linux OS based IoT devices and aims at gaining shell access of the devices to divert their operations towards the benefit of the *Mirai botnet* [7]. This kind of captured devices are used by the Mirai botnet afterwards in performing joint DDoS attacks toward more advanced targets [17]. *Torii botnet*, is a more sophisticated and an advanced version of Mirai, which also needs to be paid attention while securing industrial networks [31].

Pre-Print

2.5 How can we enforce security and privacy for industrial networks?

Given the requirements discussed in Section 2.3 and the lessons learned from the attacks listed in Section 2.4 we now list important general steps to increase the cyber defense.

1. The very first step in increasing the cyber defenses of the industrial networks is executing an extensive security risk analysis of the existing infrastructure,

including the research of software, hardware, and communication processes. Furthermore, as intrusions themselves can also provide valuable information, it would be beneficial to analyze system logs and other records of their nature and timing. Already known common weaknesses include poor code quality, improper authentication, and weak firewall rules.
2. Once the first step is completed, then it is suggested to complete an analysis of the potential consequences of the aforementioned failures or shortcomings. This includes both immediate consequences as well as second- and third-order cascading impacts on parallel systems.
3. Thirdly, risk mitigation solutions, which may include simple remediation of infrastructure inadequacies or novel strategies, can be deployed to address the situation. Some such measures include re-coding of control system algorithms to make them more capable of resisting and recovering from cyber-attacks or preventative techniques that allow more efficient detection of unusual or unauthorized changes to data. Strategies to account for human error which can compromise systems include educating those who work in the field to be wary of suspicious USB drives as in the case of Stuxnet, which can introduce malware if inserted, even if just to check their contents [39].
4. Finally, as a complementary element to the whole security architecture, an **IDS** should be employed, so that in any case of intrusion, system managers can be aware of the threats and take early action against them.

Especially the last point is of interest to us. Even though research of intrusion detection has been ongoing for over 40 years, the current design of such systems may not be optimal for industrial networks. As such, we envision that data streaming (discussed next) can be leveraged for more effective designs (Section 5).

3 Introduction to data streaming

We overview in this section the data streaming paradigm. We first discuss its differences from traditional database-based (DB-based) analysis approaches. Subsequently, we introduce basic concepts about the paradigm. Finally, we list some of the available streaming-based analysis frameworks that could be directly leveraged by security applications for industrial systems.

3.1 From Database Management Systems to Stream Processing Engines

Database Management Systems (DBMSs) have been used for decades to persist and query data. The amount of data collected in large distributed systems, which has always increased since the introduction of the first pioneer DBMSs, motivated research focusing on distributed and parallel storage and processing. The popularity

Pre-print

of DBMSs is not only due to their efficient managing of data, but also due to their powerful language (e.g., SQL), that allows for complex processing of data by means of a set of well known basic commands.

Since the year 2000, nonetheless, the emergence of IoT setups and sensor networks, together with the explosion of the amounts of sensed data, started showing the limitations of DBMSs. Such limitations stem from the fact that the primary goal of DBMSs is data persistence rather than data querying. That is, they are designed to efficiently maintain data that is accessed and aggregated only when a query is issued by a user (or accessed by queries triggered based on user-defined conditions). This data processing paradigm incurs high overheads when applied to applications that are mainly designed to transform and aggregate raw data (possibly coming from unbounded streams) into small, manageable sets of data.

(a) DBMSs information processing. (b) SPEs information processing.

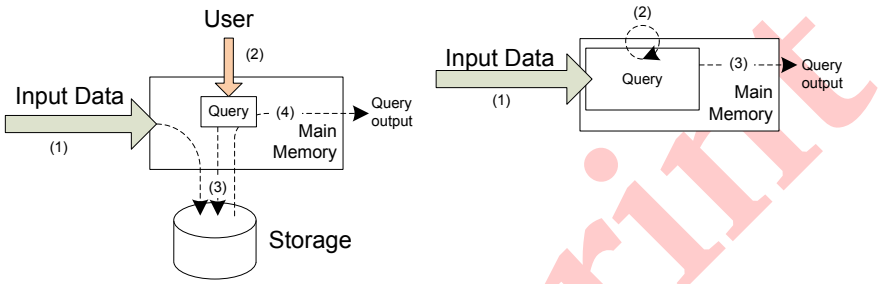

Fig. 2: Information processing overview for DBMSs (a) and SPEs (b) [21].

To better understand the limitations of DBMSs, Figure 2a (from [21]) presents a high-level overview of how data is stored and processed by a DBMS. Input data is initially persisted into *relations* (1). When a request to process a query from a user is received (2), the query is instantiated and data is read from the storage (3), processed and delivered as output (4). This approach, sometimes referred to as *first the data, then the query* incurs the unnecessary overhead of storing incoming data for applications that are interested in the results of the analysis but that do not require the input data to be persisted once such results are produced.

When the data streaming paradigm was first introduced [40], the main idea was to change the architecture of traditional DBMSs by removing the persistence of each incoming message. The removal of the persistence reduces the per-tuple processing latency significantly as writes to and reads from persistent storage take significantly longer than accesses to main memory.

However, such a modification introduces several new challenges about how data is processed, such as the fact that the available memory is smaller than the available storage space; hence, portions of raw data can only be maintained during limited periods of time. Figure 2b (from [21]) overviews how a Stream Processing Engine

Pre-print

(SPE) processes data. Input information is processed directly by a continuous query (1). For each incoming message, the query updates its internal state (2). Finally, if it is the case, an output is generated by the continuous query (3).

3.2 Basic concepts and sample application

A streaming *continuous query* (or simply query, in the remainder) consists of streams and operators. A stream is an unbounded sequence of tuples sharing a schema $\langle ts, a_1, \ldots, a_n \rangle$ where ts is the *timestamp* of the tuple and carries the notion of time for the queries later processing such tuples, and $a_1, \ldots, a_n$ are application-related *attributes*. In a query, tuples are processed by a Directed Acyclic Graph (DAG) of *operators*, which can also produce new tuples and deliver results to data analysts.

The common operators provided by SPEs includes *Aggregate*, *Join*, *Stateless* and *Merge* operators [24, 21]. Aggregate operators apply aggregation functions over *sliding windows* of tuples. Windows are defined by their size, their advance and, optionally, by a group-by parameter referring to one or more of the input tuples' attributes when the aggregation function is applied independently to each group of tuples sharing such attributes. The Join operator matches tuples from two streams (keeping a sliding window for each stream) and forwards the pairs for which a given predicate holds. Stateless operators, as the name suggests, do not maintain a state evolving with the tuples being processed, and can produce zero, one or more output tuples for each input tuple, applying a user-defined function that specifies the input tuples' attributes to be copied to the output tuples and the functions to be applied to them. Finally, Merge operators allow to merge multiple streams into a single one.

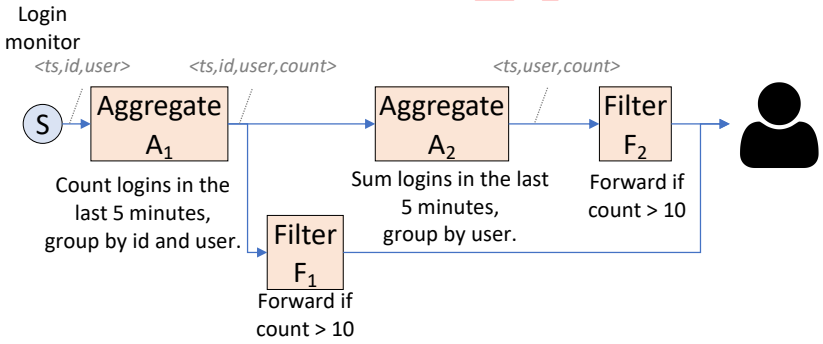

Fig. 3: Sample query monitoring the number of login attempts over a distributed set of terminals and generating an alert if more than 10 attempts are made for the same user (either from one or multiple terminals) over a period of 5 minutes.

Figure 3 shows a sample query intended to monitor the number of login attempts at the terminals of an industrial setup and generates an alert if such number is suspiciously high. More concretely, the application should generate an alert if more than 10 login attempts are made by a user over a period of 5 minutes. It should be noticed that the suspicious number of attempts can be generated from one as well as from many terminals, and the query should generate outputs that allow to differentiate between the two cases.

The query is composed by Aggregate and Filter operators. In the example, each login monitor is a source of data and generates a tuple composed by attributes ts, id and user (the schema of each stream is shown in grey above each stream). Each tuple specifies which terminal *id* has been used to attempt a login from a given *user* at a given time *ts*. Aggregate A_1 counts the number of distinct login attempts on a per-id, per-user basis, and forwards such count both to Filter F_1 and to Aggregate A_2. Filter F_1 forwards suspicious tuples to the analyst. At the same time, aggregate A_2 produces the cumulative count on a per-user basis. Finally, Aggregate A_2's output tuples are forwarded to filter F_2 which, based on the count, discards them or forwards them to the analyst. The latter can distinguish whether a suspicious alert has been generated from a single terminal or many of them depending on the output stream delivering such alert.

3.3 Commonly used Stream Processing Engines

SPEs have rapidly evolved from research prototypes such as Aurora [2] and Borealis [1] to solutions leveraged by many tech companies. Among existing ones, the most widely adopted include Apache Flink [16], Apache Storm [41], Apache Kafka [26], Apache Beam [9] and Twitter's Heron [25]. While they differ in their architectural choices and the specific APIs they make available to programmers, they usually provide the aforementioned common set of operators for programmers to compose streaming-based applications.

4 The Role of Intrusion Detection Systems (IDSs)

Before discussing the special needs of intrusion detection for industrial networks, we will give an overview of the state-of-the art systems used for normal IT systems. As such, Section 4.1 introduces layered cyber-security life cycle of information systems. IDS categories are introduced in Section 4.2 and especially anomaly-based IDSs. Specific traditional IDSs that have been used in industrial networks are then summarized in Section 4.3.

Pre-print

4.1 Layered Cyber-Security Life Cycle of Information Systems

As discussed in [11], in order to provide a complete solution against cyber-attacks, any cyber-security system should have a layered defense structure as shown in Figure 4, consisting of prevention, detection and mitigation layers.

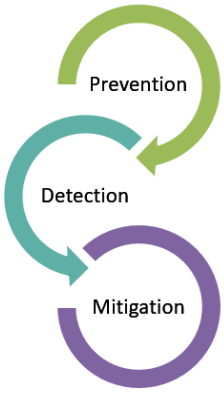

Fig. 4: Layered Cyber-Security Life Cycle of Information Systems [14]

1. **Prevention:** This layer, when employed as a system, is referred to as an Intrusion Prevention System (IPS) and constitutes the first line of defense against intrusions. Sometimes, IPSs such as firewalls are not fully trusted and/or they are not efficient enough to prevent all types of attacks towards our networks.
2. **Detection:** This layer, when employed as a system, is also referred to as Intrusion Detection System (IDS) and constitutes the second line of defense against intrusions. History has taught us that the first line of defense, IPSs, may fail as in the case of many severe incidents of cyber-attacks against critical infrastructures such as nuclear enrichment facilities, electric grid, etc., as discussed earlier in Section 2.4. IDSs are complementary for system administrators by offering further solutions to the problem by detecting intrusions to their network on time, so that the threats can be mitigated. Therefore, IDSs are as important as IPSs.
3. **Mitigation:** The last line of defense in the cyber-security is the mitigation, which includes security measures (such as disabling some ports, limiting the Internet access, etc.) to be taken after an intrusion incident is detected.

Even though the methodologies of Prevention / Detection / Mitigation is well-founded in the literature, the terms of IPS and IDS sometimes shift. As an example, we can take Snort (an open-source, free and lightweight network IDS [38]), the de-facto standard when it comes to signature-based IDSs. It started out as an IDS to

Pre-print

detect attacks, but later developments allowed it to also react (mitigate) attacks. As such, it is now classified as an intrusion detection and prevention system.

Looking at more specific systems such as industrial networks, vulnerabilities, cyber attacks and the need for IDSs are stressed by Butun et al. [14]. Accordingly, IDS is a very important part of any cyber-security system, and in many cases bears the weight of the overall cyber-defense.

4.2 Intrusion Detection Systems (IDSs)

As stated earlier in Section 4.1, Intrusion Detection Systems (IDSs) will constitute the second line of defense against intrusions towards industrial networks. Therefore, it is important to learn their types along with their working principles. In computer science, IDSs can be classified into three categories according to their detection methodology [18, 11, 13]:

1. Anomaly-based IDS,
2. Misuse (signature)-based IDS,
3. Specification-based IDS.

The advantages and disadvantages of IDS types are as shown in Table 1.

Table 1: Comparison of IDS types.

IDS Type	**Advantages**	**Disadvantages**
Anomaly detection -based	- can handle unknown attacks - does not need frequent updates - easy to configure/generalize	- low accuracy - high false positive ratio - high false negative ratio
Misuse detection -based	- high accuracy - low false positive/negative ratios	- can not handle unknown attacks - need frequent updates
Specification detection -based	- high accuracy - cost-efficient - low false positive/negative ratios	- hard to design - hard to generalize

Commonly, misuse-based systems manually encode indicators of attacks, so-called signatures. As such, these systems are quite specific when they alert in that they can give the type of attack the system is exposed to. These systems work well with very well and categorized attacks of the past. However, they are quite useless in the case of new attack vectors that can not be specified with the old ones.

Specification-based systems are built on the specifications of the allowed behavior of the system and many times used for network protocols. Such systems can be successful when a formal description of the system behavior exists, and that this document is strictly followed. They have been suggested for network protocols, but for very complex systems, a formal specification often does not exist or the behavior of the system is quite dynamic. Specification-based IDS is reported as

Pre-Print

more suitable to detect process-based attacks, and on the contrary, is observed to be typically expensive for large deployments (such as factory environments) to set up and comparably less scalable [20].

Anomaly-based system are often built using machine-learning techniques in that they try to model the normal system behavior (regardless of its formal specifications). When the current behavior is different *enough* from the learned profile, an alert is generated. Opposed to misuse-based systems, anomaly-based systems are much less exact as they only alert for system anomalies and not true attacks. On the other hand, anomaly-based systems may be able to alert for future unknown attacks (zero-day attacks) if they cause a visible effect in the monitored data [14].

As discussed in [44], it is claimed that the best approach of IDS for industrial networks is *Anomaly-based IDS*, due to its capability of detecting unknown and new types of attacks. Moreover, many industrial networks are quite regular in their behavior (M2M communication), meaning that some of their weaknesses are less pronounced when deployed in industrial networks.

A taxonomy of anomaly detection-based IDSs is shown in Figure 5, and consists of mainly the following approaches: statistical, data mining and Artificial Intelligence (AI) [14]. Each of these approaches is then sub-classified into various methods. The interested reader is referred to [13] for a detailed discussion of these types.

4.3 Examples of traditional IDS deployed for industrial networks

This section presents two recent works focusing on IDS for industrial networks. These approaches employ anomaly-based techniques in an effort of detecting intrusions toward industrial networks but they do not utilize the streaming paradigm to efficiently process data.

4.3.1 Example #1

A recent study by Anton *et al.* [6] on industrial networks has used two different solutions for anomaly detection to capture intrusions:

1) Artificial Intelligence/SVM -based solution: This solution is based on the *Support Vector Machine (SVM)* which is a machine learning-based anomaly detection method. The *SVM* is a supervised classification and regression analysis method in machine learning which is equipped with learning algorithms to analyze any separated set of data used for classification, meaning the training set needs to contain enough information so that the examples of the different categories are divided by a clear gap that is separating the two groups as wide as possible. New samples are then mapped into the same domain and categorized based on the proximity to the which group they fall near by. It is a large margin classifier and it is trained with a labeled set of instances. The training session is followed by another session, in which the identification and classification of the test and productive data are performed.

Pre-print

2) Data mining/Data clustering -based solution: This solution is based on the *Random Forest* which is a data mining/data clustering and outlier detection-based anomaly detection method. *Random Forest* is a collection of *Decision Trees* which have many binary classifiers comprised of internal split leaf nodes used to classify events and bundled together to a one root node. The *majority voting* of all decision trees is the main classification of the Random Forest, hence they are robust to over-shooting of the data and can converge to the intended best fit quickly, making them applicable in a variety of use case scenarios [6].

The data containing industrial network in operation is analysed for the sake of discovering the attacks targeted the data. As shown in Table 2, two different data-

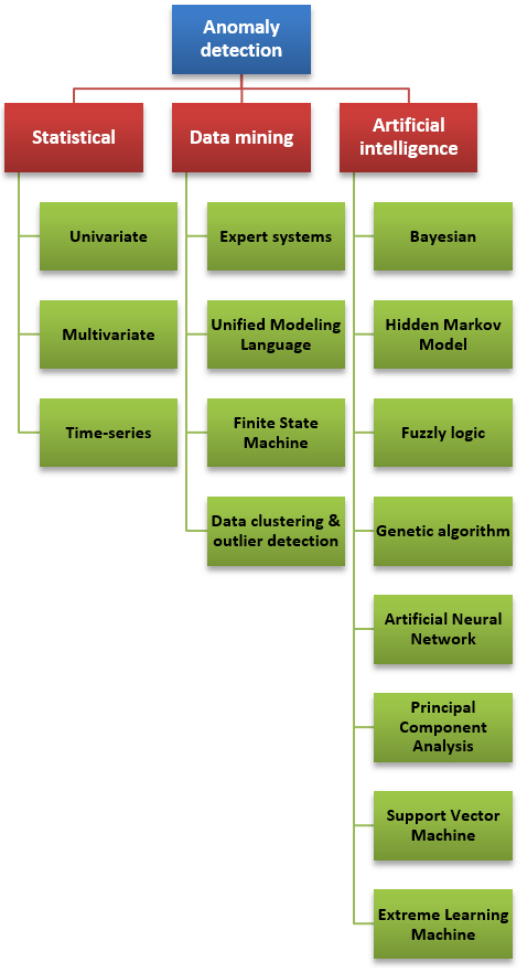

Fig. 5: Taxonomy of Anomaly Detection-based IDSs [14]

sets are employed: Modbus-based gas pipeline control traffic and Object Linking and Embedding for Process Control Unified Architecture (OPC UA)-based batch processing traffic data (from the tanks, pumps, actuators and sensors consisting of water level, flow volume, pressure, temperature, and pump status).

Table 2: Parameters of the used data sets in [6].

ID	Protocol	# of packets	Duration	Attacks	# of mal. pac.	% of mal. pac.
DS1	Modbus	274,627	4 days	none	60,048	22
DS2	OPC UA	4,910	41 minutes	2	702	14

mal. pac.: malicious packets

3) Comparison: According to the authors, both anomaly detection methods performed well, with the Random Forest method slightly outperforming the SVM method. Methods of machine learning (such as SVM) and data mining (such as Random Forests) can enhance the detection rate of the commercial IDSs in the market. These approaches benefit from a limited state set of the systems as well as a large amount of training data for a given environment, as industrial environments are fruitful data generators.

4.3.2 Example #2

In [8], Aoudi et al. introduce the notion of departure-based attack detection where a departure is a specific type of anomaly that refers to the process dynamics being forced to depart from the normal behavior due to potentially malicious structural changes in the stream of sensor measurements. The normal behavior is established in an offline training phase through a mathematical construction that enables the representation of the process dynamics in a noise-reduced geometric space. Thereafter, to detect a departure in the process behavior, devised method computes a departure score during an online detection phase in an iterative way. Whenever this departure score crosses a predetermined threshold, an alarm is raised to the operators.

Aoudi *et al.* argued that their devised specification-agnostic technique can successfully defend industrial networks against by detecting *stealthy attacks* (see Section 2.4). This is achieved by monitoring time-series of sensor measurements in the industrial network for structural changes in their behavior.

4.3.3 Summary

These two recent examples from the literature show very promising results in detecting attacks in industrial networks. The first example with two methods are using network traffic as input data to find the attacks. As such, a specific wiretap needs to be installed somewhere in Figure 1. If several devices communcicate over the

Pre-print

same network link, a single instance may monitor several devices. Otherwise many such taps need to be installed throughout the network. The PASAD algorithm by Aoudi et al. (example 2) use sensor data from the process as the input to be analyzed for devations. As such, it is favorably installed locally to each device. However, the resulting alerts should then be propagated through the system so larger-scale attacks can be detected.

As we discuss in the next section, data streaming for intrusion detection will naturally allow a flow of information throughout the system, making the deployment of example 1 easier, and allow systems such as described in example 2 to easily communicate between nodes.

5 IDS and Data Streaming

We present in this section why (and how) streaming-based applications can be leveraged for efficient intrusion detection in industrial setups. We begin discussing the different deployment options for the analysis of data in industrial setups. Then, we show how data streaming can enable such deployments by decoupling the semantics of security applications from their distributed and parallel execution. To exemplify this, we also extend the example introduced in Section 3. Finally, we present some of the streaming-based IDSs discussed in the literature.

5.1 What deployment options exist for the data analysis?

We can identify three possible hierarchy levels at which data can be processed within an industrial network. In the context of a smart grid, for instance, the options may include following (refer to the details presented in Section 2.2):

- At the smart meters,
- at the data concentrator,
- at the AMI head-end.

It should be noted that these options are not mutually exclusive. That is, a security application could run both at the meters as well as at the concentrators, as also discussed in [22].

Table 3 is inspired from [19] and provides a good comparison of all entities in the smart grid systems, especially from the data streaming point of view. As we travel from the smart meter to the AMI head-end (leaf to the root), the amount of data to be handled increases drastically, as well as the speed required to carry and process that data. This is a clear indication of where the data streaming can be used and why.

Pre-print

Table 3: Comparison of AMI entities [19].

Property	Smart meter	Data concentrator	AMI head-end
Amount of data	Small volume, as data sources are customer's HAN and its associated devices	Large volume, comparatively as it has to handle data from about a few hundred to tens of thousands of smart meters	Huge volume (Big Data), as it has to process data from about several millions of smart meters
Resources*	in KB range, are very restrictive	in MB range, more powerful	in GB/TB range, plentiful resources composed of high-end servers
Data speed	Comparatively low, because of non-frequent requests at the smart meter	High, as it aggregates a good number of smart meters' data	Very high, as it needs to handle a huge amount of meter data, event data, commands, etc.

*main memory, processor capacity, etc.

5.2 Leveraging distributed and parallel execution of streaming applications in industrial setups

As we discussed in Section 3, a streaming application is a DAG of base operators composed into a continuous query. Once the semantics of a certain streaming application have been expressed by composing such base operators, parallel execution is achieved by (1) instantiating multiple copies of each operator and enabling data-parallelism by distributing their input data (according to the semantics of the operator) and then collecting the results, as well as by (2) assigning such operators to distinct threads (possibly belonging to different processes and/or nodes).

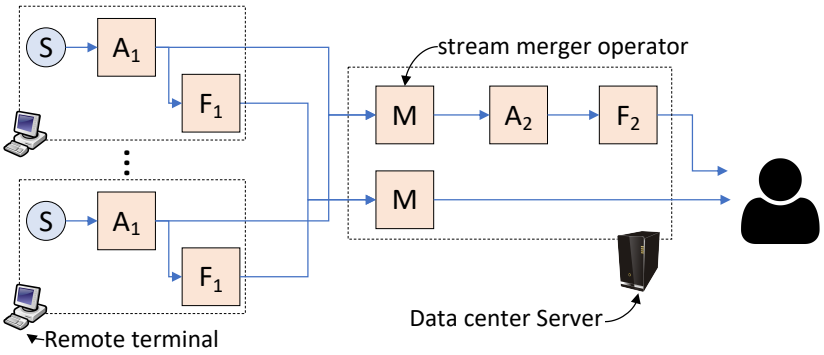

Fig. 6: Distributed and parallel deployment of the sample query from Figure 3. The query monitors the number of login attempts over a distributed set of terminals and generates an alert if more than 10 attempts are made for the same user (either from one or multiple terminals) over a period of 5 minutes.

Figure 6 shows a possible deployment plan for the sample query introduced in Figure 3. In this example, we assume an industrial setup is composed of a collection of remote terminals and a set of servers that, together, compose a data center used for the analysis of gathered data. The idea is to run the query presented in Figure 3, which monitors the number of login attempts over a distributed set of terminals and generates an alert if more than 10 attempts are made for the same user (either from one or multiple terminals) over a period of 5 minutes, in a parallel and distributed fashion, making the overall analysis scalable by distributing as much as possible data at the edge rather than gathering all of it in the central data center. In the sample deployment, a pair of operators A_1 and F_1 are deployed at each remote terminal to locally check if more than 10 login attempts are made by each user. If that is the case, the output produced by operator F_1 (which is merged by a merger operator M at a different node that, in this example, is a server at a central data center) is then forwarded to the end user. At the same time, the partial counts forwarded by all instances of operator A_1 are also merged at the data center server and the cumulative sums, if exceeding the threshold 10, are also forwarded as alerts.

Leveraging of the data streaming paradigm and SPEs allows for monitoring applications to be deployed at all the levels of a given industrial setup hierarchy by means of distributed and parallel operator execution [33, 30]. Distributed execution (achieved by means of *inter-operator parallelism*) allows for operators belonging to the same query to be run at different nodes (e.g., A_1 and A_2 in Figure 6). At the same time, parallel execution (achieved by means of *intra-operator parallelism*) allows for individual operators to be run in parallel at arbitrary numbers of nodes (e.g., A_1 and F_1 in Figure 6). We refer the reader to [21] for a exhaustive discussion about the parallelization of data streaming operators.

5.3 Correctness guarantees enabled by the data streaming processing paradigm

The execution of a query is said to be *deterministic* if feeding the query the same sequence of input tuples results in the query producing the same sequence of output tuples. For monitoring applications in CPSs, determinism is crucial to achieve correct and predictable/repeatable analysis, especially when the latter is used to generate sensitive or safety-related alerts [32].

A way to achieve such behavior is to compose queries with base operators that, if fed the same sequence of input tuples, produce the same sequence of output tuples [21, 46]. That is, operators that have no randomness in their analysis and whose analysis does not depend on execution-specific aspects [21, 4]. The common way for stateful operators to achieve deterministic execution is to base the latter on the notion of time carried by the tuples' timestamps (also referred to as *event* time [16, 4])[2]. Continuing the example of Figure 6, this implies that, once each

[2] In this case, a common assumption in the literature is that all nodes of a distributed setup have synchronized clocks [24].

pre-print

remote terminal generates a tuple about a user login and sets a timestamp to it, an alert will be generated (if the threshold condition is satisfied) independently of the latency and interleaving of messages at the data center server.

5.4 Applications of the streaming paradigm in the context of IDSs in the literature

Even though streaming has been developed since 2000 and offers a structured way to consider parallel and distributed analysis of data over complex systems such as the smart grid [23, 10, 35, 34], few examples of intrusion detection systems that use the technique exists in the literature. Below we summarize two promising examples to highlight the possibilities.

METIS

Gulisano et al. [22] propose METIS. METIS relies on probabilistic models for detection and is designed to detect challenging attacks in which the adversaries aim at going unnoticed. Owing to its two-tier architecture, it eases the modeling of possible adversary goals and allows for a fully distributed and parallel traffic analysis through the data streaming processing paradigm. At the same time, it allows for complementary intrusion detection systems to be integrated in the framework. Gulisano *et al.* have shown METIS' use and functionality through an energy exfiltration attack scenario, in which an adversary aims at stealing energy information from AMI users. Based on a prototype implementation using the Apache Storm SPE and a very large dataset from a real-world AMI, authors have shown that METIS is not only able to detect such attacks, but that it can also handle large volumes of data even when run on commodity hardware.

Stream Data Mining for Distributed-IDS

Alseiari et al. [5] propose a real-time Distributed Intrusion Detection System (DIDS) for the AMI infrastructure that utilizes stream data mining techniques with a multi-layer implementation approach. Using unsupervised online clustering technique, the anomaly-based DIDS monitors the data flow in the AMI and distinguishes if there is anomalous traffic. The authors employed *Mini-Batch K-means* clustering technique which is a variant of "K-means" clustering algorithm, that is suited to data stream models.

It incrementally fits the new sessions to the existing model by continuously learning and updating the clusters in real time which significantly reduces running and computation time. However, K-means follows a static procedure by discarding the previously created clusters and building a new model from scratch in every *sliding*

Pre-print

window iteration, hence using more memory and time. By executing a comparison between online and offline clustering techniques, the authors claim that their experimental results have shown that online clustering "Mini-Batch K-means" algorithm was able to suit the architecture requirements by giving high detection rate and low false positive rates.

5.5 Security Implications and Opportunities of Data Streaming

As in every complex system, inclusion of new components (as well as removal of the old ones) might provide opportunities along with implications. As such, inclusion of *data streaming* paradigm will have pros and cons against the already existing components, users, etc, as discussed below.

Security Implications

This is a debated issue, whether centralize v.s. distributed detection of intrusions is more critical and beneficial. It is argued in this chapter that distributed and real-time monitoring of events (intrusions) is important for industrial networks, especially for mission-critical systems, such as smart grid systems presented in Section 2.2, hence timely taken decisions might save people from catastrophic events.

In most instances, bottlenecks (the data traffic gets congested!) occur in the upstream path of the centralized networks. However, distributed networks are resilient to that problem. For instance, intrusion detection solutions (especially related to the data streaming) discussed in this chapter, aim at pushing the data analysis towards peripheral devices instead of gathering data centrally. This kind of solution is offered to provide a fast response by removing the necessity of the round-trip messages in between the smart meter and the server. Besides, data streaming is also beneficial to the overall network performance in distributed networks due to the decreased load in preventing the aforementioned bottlenecks.

On the other hand, centralized solutions have shown to be more effective against more dedicated and distributed attacks such as DoS and DDoS attacks.

Opportunities

This chapter argued and also discussed that the *data streaming* paradigm can be really helpful for industrial networks in fulfilling detection of intrusions timely manner, hence it allows continuous monitoring of events in an autonomous and adaptive way. It can be also beneficial in decreasing the bottlenecks in big industrial networks such as smart grid systems presented in Section 2.2, by not only enhancing the response time of the queries/commands but also decreasing the overall traffic of the network.

Pre-print

6 Conclusions

This chapter presented key points of industrial networks, especially network architecture, data handling (centralized vs. distributed), and cyber-security vulnerabilities, attacks and their counter measures. Besides, data stream processing as means to analyze data in IIoT infrastructures is described. Overall research challenges are also presented regarding cyber-security aspects of data streaming for industrial networks, especially for smart grid systems. Finally, applicability of data streaming to IDSs is proven by the evidence from the literature that appeared in the recent years.

Based on the recent developments in the field, it is concluded in this chapter that the *data streaming* concept can be utilized for industrial networks, especially for smart grid systems, by leveraging cloud and/or fog based deployments. More importantly, *data streaming* is projected to be very useful and handy for detecting intrusions in timely manner and cost efficient way.

Acknowledgements This research has been partially supported by the Swedish Civil Contingencies Agency (MSB) through the projects RICS, by the EU Horizon 2020 Framework Programme under grant agreement 773717, and by the Swedish Foundation for International Cooperation in Research and Higher Education (STINT) Initiation Grants program under grant agreement IB2019-8185.

Pre-print

List of Abbreviations

AI: Artificial Intelligence
AMI: Advanced Metering Infrastructure
CPS: Cyber-Physical Systems
DBMS: Data-Base Management System
DoS: Denial-Of-Service
DDoS: Distributed DoS
DIDS: Distributed IDS
HMI: Human-Machine Interface
IDS: Intrusion Detection Systems
IoT: Internet of Things
IIoT: Industrial Internet of Things
IPS: Intrusion Prevention System
MITM: Man-In-The-Middle
OPC UA: Object Linking and Embedding for Process Control Unified Architecture
PII: Personal Identification Information
RAT: Remote Access Trojan
RTU: Remote Terminal Unit
SCADA: Supervisory Control And Data Acquisition
SPE: Stream Processing Engine
SVM: Support Vector Machine
VPN: Virtual Private Network

References

1. Abadi, D.J., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J.H., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., et al.: The design of the borealis stream processing engine. In: CIDR, vol. 5, pp. 277–289 (2005)
2. Abadi, D.J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: a new model and architecture for data stream management. the VLDB Journal **12**(2), 120–139 (2003)
3. Abdallah, A., Shen, X.: Security and privacy in smart grid. Springer (2018)
4. Akidau, T., Bradshaw, R., Chambers, C., Chernyak, S., Fernández-Moctezuma, R.J., Lax, R., McVeety, S., Mills, D., Perry, F., Schmidt, E., et al.: The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. Proceedings of the VLDB Endowment **8**(12), 1792–1803 (2015)
5. Alseiari, F.A.A., Aung, Z.: Real-time anomaly-based distributed intrusion detection systems for advanced metering infrastructure utilizing stream data mining. In: 2015 International Conference on Smart Grid and Clean Energy Technologies (ICSGCE), pp. 148–153. IEEE (2015)
6. Anton, S.D.D., Sinha, S., Schotten, H.D.: Anomaly-based intrusion detection in industrial data with svm and random forests. arXiv preprint arXiv:1907.10374 (2019)
7. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., et al.: Understanding the mirai botnet. In: USENIX Security Symposium, pp. 1092–1110 (2017)
8. Aoudi, W., Iturbe, M., Almgren, M.: Truth will out: Departure-based process-level detection of stealthy attacks on control systems. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 817–831. ACM (2018)
9. Apache Beam. `https://beam.apache.org/`. Accessed: 2019-09-25
10. Botev, V., Almgren, M., Gulisano, V., Landsiedel, O., Papatriantafilou, M., van Rooij, J.: Detecting non-technical energy losses through structural periodic patterns in ami data. In: 2016 IEEE International Conference on Big Data (Big Data), pp. 3121–3130. IEEE (2016)
11. Butun, I.: Prevention and detection of intrusions in wireless sensor networks. University of South Florida, Ph.D. thesis (2013)
12. Butun, I.: Privacy and trust relations in internet of things from the user point of view. In: 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1–5. IEEE (2017)
13. Butun, I., Morgera, S.D., Sankar, R.: A survey of intrusion detection systems in wireless sensor networks. IEEE communications surveys & tutorials **16**(1), 266–282 (2013)
14. Butun, I., Österberg, P.: Detecting intrusions in cyber-physical systems of smart cities: Challenges and directions. In: Secure Cyber-Physical Systems for Smart Cities, pp. 74–102. IGI Global (2019)
15. Butun, I., Österberg, P., Song, H.: Security of the internet of things: Vulnerabilities, attacks and countermeasures. IEEE Communications Surveys & Tutorials (2019)
16. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., Tzoumas, K.: Apache flink: Stream and batch processing in a single engine. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering **36**(4) (2015)
17. Chaudhary, S.: Privacy and security issues in internet of things. International Education and Research Journal **3**(5) (2017)
18. Debar, H., Dacier, M., Wespi, A.: Towards a taxonomy of intrusion-detection systems. Computer Networks **31**(8), 805–822 (1999)
19. Faisal, M.A., Aung, Z., Williams, J.R., Sanchez, A.: Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study. IEEE Systems journal **9**(1), 31–44 (2014)
20. Fauri, D., Dos Santos, D.R., Costante, E., den Hartog, J., Etalle, S., Tonetta, S.: From system specification to anomaly detection (and back). In: Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy, pp. 13–24. ACM (2017)

Pre-print

21. Gulisano, V.: Streamcloud: an elastic parallel-distributed stream processing engine. Ph.D. thesis, Universidad Politécnica de Madrid (2012)
22. Gulisano, V., Almgren, M., Papatriantafilou, M.: Metis: a two-tier intrusion detection system for advanced metering infrastructures. In: International Conference on Security and Privacy in Communication Networks, pp. 51–68. Springer (2014)
23. Gulisano, V., Almgren, M., Papatriantafilou, M.: When smart cities meet big data. Smart Cities **1**(98), 40 (2014)
24. Gulisano, V., Jimenez-Peris, R., Patino-Martinez, M., Soriente, C., Valduriez, P.: Streamcloud: An elastic and scalable data streaming system. IEEE Transactions on Parallel and Distributed Systems **23**(12), 2351–2365 (2012)
25. HERON: A realtime, distributed, fault-tolerant stream processing engine from Twitter. https://apache.github.io/incubator-heron/. Accessed: 2019-09-30
26. Kafka Streams. https://kafka.apache.org/documentation/streams/. Accessed: 2019-10-09
27. Kumar, P., Lin, Y., Bai, G., Paverd, A., Dong, J.S., Martin, A.: Smart grid metering networks: A survey on security, privacy and open research issues. IEEE Communications Surveys & Tutorials (2019)
28. Mo, Y., Kim, T.H.J., Brancik, K., Dickinson, D., Lee, H., Perrig, A., Sinopoli, B.: Cyber–physical security of a smart grid infrastructure. Proceedings of the IEEE **100**(1), 195–209 (2011)
29. Mo, Y., Sinopoli, B.: On the performance degradation of cyber-physical systems under stealthy integrity attacks. IEEE Transactions on Automatic Control **61**(9), 2618–2624 (2015)
30. Najdataei, H., Nikolakopoulos, Y., Papatriantafilou, M., Tsigas, P., Gulisano, V.: Stretch: Scalable and elastic deterministic streaming analysis with virtual shared-nothing parallelism. In: Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems, pp. 7–18. ACM (2019)
31. Osborne, C.: Meet torii, a new iot botnet far more sophisticated than mirai variants. `https://www.zdnet.com/article/meet-torii-a-new-iot-botnet-far-more-sophisticated-than-mirai/` (2018). Accessed: 2018-10-09
32. Palyvos-Giannas, D., Gulisano, V., Papatriantafilou, M.: Genealog: Fine-grained data streaming provenance at the edge. In: Proceedings of the 19th International Middleware Conference, pp. 227–238. ACM (2018)
33. Palyvos-Giannas, D., Gulisano, V., Papatriantafilou, M.: Haren: A framework for ad-hoc thread scheduling policies for data streaming applications. In: Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems, pp. 19–30. ACM (2019)
34. van Rooij, J., Gulisano, V., Papatriantafilou, M.: Locovolt: Distributed detection of broken meters in smart grids through stream processing. In: Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems, pp. 171–182. ACM (2018)
35. van Rooij, J., Swetzén, J., Gulisano, V., Almgren, M., Papatriantafilou, M.: echidna: Continuous data validation in advanced metering infrastructures. In: 2018 IEEE International Energy Conference (ENERGYCON), pp. 1–6. IEEE (2018)
36. Sayfayn, N., Madnick, S.: Cybersafety analysis of the maroochy shire sewage spill, working paper cisl# 2017-09. Cybersecurity Interdisciplinary Systems Laboratory (CISL), Sloan School of Management, Massachusetts Institute of Technology pp. 2017–09 (2017)
37. Sharma, R.: How does gdpr affect smart grids? https://www.energycentral.com/c/iu/how-does-gdpr-affect-smart-grids (2018)
38. Snort: Snort network intrusion detection system. `https://www.snort.org`. Accessed: 2019-11-05
39. Sridhar, S., Hahn, A., Govindarasu, M.: Cyber–physical system security for the electric power grid. Proceedings of the IEEE **100**(1), 210–224 (2011)
40. Stonebraker, M., Çetintemel, U., Zdonik, S.: The 8 requirements of real-time stream processing. ACM Sigmod Record **34**(4), 42–47 (2005)
41. Apache Storm. http://storm.apache.org/ (2017)
42. Stylianopoulos, C.: Parallel and distributed processing in the context of fog computing: High throughput pattern matching and distributed monitoring. Licentiate Thesis, Chalmers University of Technology (2018)

Preprint

43. Tesfay, T.T.: Cybersecurity solutions for active power distribution networks. Ph.D. thesis, EPFL (2017)
44. Tong, W., Lu, L., Li, Z., Lin, J., Jin, X.: A survey on intrusion detection system for advanced metering infrastructure. In: 2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), pp. 33–37. IEEE (2016)
45. Vasserman, E.Y., Hopper, N.: Vampire attacks: draining life from wireless ad hoc sensor networks. IEEE transactions on mobile computing **12**(2), 318–332 (2011)
46. Walulya, I., Palyvos-Giannas, D., Nikolakopoulos, Y., Gulisano, V., Papatriantafilou, M., Tsigas, P.: Viper: A module for communication-layer determinism and scaling in low-latency stream processing. Future Generation Computer Systems **88**, 297 – 308 (2018). DOI https://doi.org/10.1016/j.future.2018.05.067. URL `http://www.sciencedirect.com/science/article/pii/S0167739X17326791`
47. Yan, Q., Huang, W., Luo, X., Gong, Q., Yu, F.R.: A multi-level ddos mitigation framework for the industrial internet of things. IEEE Communications Magazine **56**(2), 30–36 (2018)

Pre-print